

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

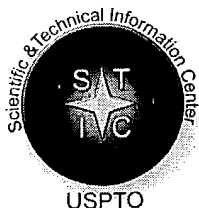
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

L Number	Hits	Search Text	DB	Time stamp
6	2	send\$1 same file\$1 same section\$1 same request\$4 same remote\$1 same network\$1 same server\$1	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM TDB	2004/09/27 20:02
7	31	status same information\$1 same section\$1 same indicat\$4 same success\$4 same (read\$4 or writ\$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM TDB	2004/09/27 20:04
8	8	(status same information\$1 same section\$1 same indicat\$4 same success\$4 same (read\$4 or writ\$4)) same (respon\$4 same return\$4)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM TDB	2004/09/27 20:05
9	0	((status same information\$1 same section\$1 same indicat\$4 same success\$4 same (read\$4 or writ\$4)) same (respon\$4 same return\$4)) and (redirector\$4 same receiv\$4 same buffer\$1)	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM TDB	2004/09/27 20:05

	Type	Hits	Search Text	DBs
1	BRS	0	send\$4 same file\$1 same remote\$1 same network\$4 same server\$1 same await\$3 same status same respons\$3	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB
2	BRS	2	send\$4 same file\$1 same remote\$1 same network\$4 same server\$1 same await\$3	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB
3	BRS	0	status\$1 same iformation\$1 same file\$1 same section\$1 same request\$1 same success\$1 same response\$1	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB
4	BRS	2	status\$1 same information\$1 same file\$1 same section\$1 same request\$1 same success\$1 same response\$1	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB
5	BRS	0	(client\$1 or node\$1) same redirector\$1 same receiv\$4 same (read\$4 or writ\$4) same remote\$1 same buffer\$4	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM_TDB

	Time Stamp
1	2004/09/27 19:12
2	2004/09/27 19:16
3	2004/09/27 19:18
4	2004/09/27 19:20
5	2004/09/27 19:22

L Number	Hits	Search Text	DB	Time stamp
1	6910550	send\$4 same file\$1 same remote\$1 same network\$4 same server\$1 same await\$3 s	USPAT; US-PGPUB; EPO; JPO; DERWENT; IBM TDB	2004/09/27 20:24
2	241443	709/203, "217"	USPÄT; US-PGPUB; EPO; JPO; DERWENT; IBM TDB	2004/09/27 20:23
3	8041	709/203, 217.ccls.	USPÄT; US-PGPUB; EPO; JPO; DERWENT; IBM TDB	2004/09/27 20:23
4	0	(709/203, 217.ccls.) and (send\$4 same file\$1 same remote\$1 same network\$4 same server\$1 same await\$3 same status same respons\$3)	USPÄT; US-PGPUB; EPO; JPO; DERWENT; IBM TDB	2004/09/27 20:24



STIC Search Report

EIC 2100

STIC Database Tracking Number: 133674

TO: Mohammad Ali
Location: 4Y18
Art Unit : 2177
Monday, September 27, 2004

Case Serial Number: 10/003137

From: Geoffrey St. Leger
Location: EIC 2100
PK2-4B30
Phone: 308-7800

geoffrey.stleger@uspto.gov

Search Notes

Dear Examiner Ali,

Attached please find the results of your search request for application 10/003137. I searched Safari Books Online and the Internet.

Please let me know if you have any questions.

Regards,



Geoffrey St. Leger
4B30/308-7800



From: ProQuest

US Patent and Trademark Office

[Home](#) [Desktop](#) [Library](#) [Recent Searches](#) [Recent Pages](#) [Notes](#) [Bookmarks](#) [Log](#)

▼ Search

☐ Current Book

☐ Code Fragments only[Advanced Search](#)

▼ Table of Contents



TCP/IP Illustrated,
Volume 1: The
Protocols



- [Table of Contents](#)

TCP/IP Illustrated, Volume 1: The Protocols

By W. Richard Stevens

Publisher : Addison Wesley
 Pub Date : December 15, 1993
 ISBN : 0-201-63346-9
 Pages : 600
 Slots : 2.0

- ☐ Copyright
- ☐ Preface
- ☐ Introduction
- ☐ Link Layer
- ☐ IP: Internet Protocol
- ☐ ARP: Address Resolution Protocol
- ☐ RARP: Reverse Address Resolution Protocol
- ☐ ICMP: Internet Control Message Protocol
- ☐ Ping Program
- ☐ Traceroute Program
- ☐ IP Routing
- ☐ Dynamic Routing Protocols
- ☐ UDP: User Datagram Protocol
- ☐ Broadcasting and Multicasting
- ☐ IGMP: Internet Group Management Protocol
- ☐ DNS: The Domain Name System
- ☐ TFTP: Trivial File Transfer Protocol
- ☐ BOOTP: Bootstrap Protocol
- ☐ TCP: Transmission Control Protocol
- ☐ TCP Connection Establishment and Termination
- ☐ TCP Interactive Data Flow

TCP/IP Illustrated, Volume 1 is a complete and detailed guide to the TCP/IP protocol suite - with an important difference from the subject. Rather than just describing what the RFCs say the suite should do, this unique book uses a popular diagnostic tool to actually watch the protocols in action.

By forcing various conditions to occur - such as connection timeout and retransmission, and fragmentation - and then results, *TCP/IP Illustrated* gives you a much greater understanding of concepts than words alone could provide. Whether you are new to the subject or you have read other books on the subject, you will come away with an increased understanding of how and why TCP/IP works and the skill at developing applications that run on it.

URL <http://proquest.safaribooksonline.com/0201633469>

User name: US Patent and Trademark Library
Book: TCP/IP Illustrated, Volume 1: The Protocols
Section: Chapter 20. TCP Bulk Data Flow

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

20.1 Introduction

In Chapter 15 we saw that TFTP uses a stop-and-wait protocol. The sender of a data block required an acknowledgment for that block before the next block was sent. In this chapter we'll see that TCP uses a different form of flow control called a *sliding window* protocol. It allows the sender to transmit multiple packets before it stops and waits for an acknowledgment. This leads to faster data transfer, since the sender doesn't have to stop and wait for an acknowledgment each time a packet is sent.

We also look at TCP's PUSH flag, something we've seen in many of the previous examples. We also look at slow start, the technique used by TCP for getting the flow of data established on a connection, and then we examine bulk data throughput.

URL <http://proquest.safaribooksonline.com/0201633469/ch20lev1sec1>

User name: US Patent and Trademark Library
Book: TCP/IP Illustrated, Volume 1: The Protocols
Section: Chapter 20. TCP Bulk Data Flow

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

20.2 Normal Data Flow

Let's start with a one-way transfer of 8192 bytes from the host `svr4` to the host `bsdi`. We run our `sock` program on `bsdi` as the server:

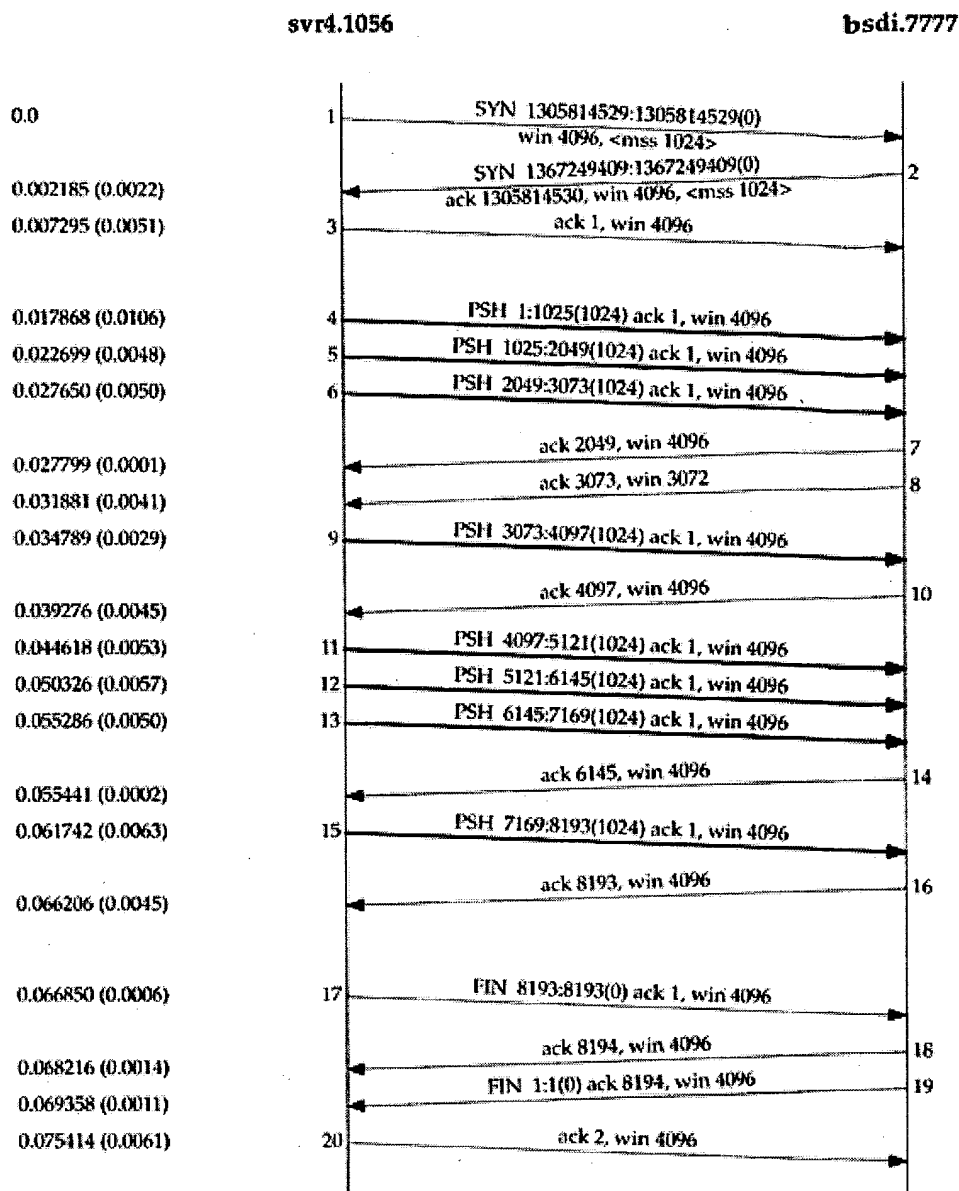
```
bsdi % sock -i -s 7777
```

The `-i` and `-s` flags tell the program to run as a "sink" server (read from the network and discard the data), and the server's port number is specified as `7777`. The corresponding client is then run as:

```
svr4 % sock -i -n8 bsdi 7777
```

This causes the client to perform eight 1024-byte writes to the network. Figure 20.1 shows the time line for this exchange. We have left the first three segments in the output to show the MSS values for each end.

Figure 20.1. Transfer of 8192 bytes from `svr4` to `bsdi`.



The sender transmits three data segments (4–6) first. The next segment (7) acknowledges the first two data segments only. We know this because the acknowledged sequence number is 2049, not 3073.

Segment 7 specifies an ACK of 2049 and not 3073 for the following reason. When a packet arrives it is initially processed by the device driver's interrupt service routine and then placed onto IP's input queue. The three segments 4, 5, and 6 arrive one after the other and are placed onto IP's input queue in the received order. IP will pass them to TCP in the same order. When TCP processes segment 4, the connection is marked to generate a delayed ACK. TCP processes the next segment (5) and since TCP now has two outstanding segments to ACK, the ACK of 2049 is generated (segment 7), and the delayed ACK flag for this connection is turned off. TCP processes the next input segment (6) and the connection is again marked for a delayed ACK. Before segment 9 arrives, however, it appears the delayed ACK timer goes off, and the ACK of 3073 (segment 8) is generated. Segment 8 advertises a window of 3072 bytes, implying that there are still 1024 bytes of data in the TCP receive buffer that the application has not read.

Segments 11–16 show the "ACK every other segment" strategy that is common. Segments 11, 12, and 13 arrive and are

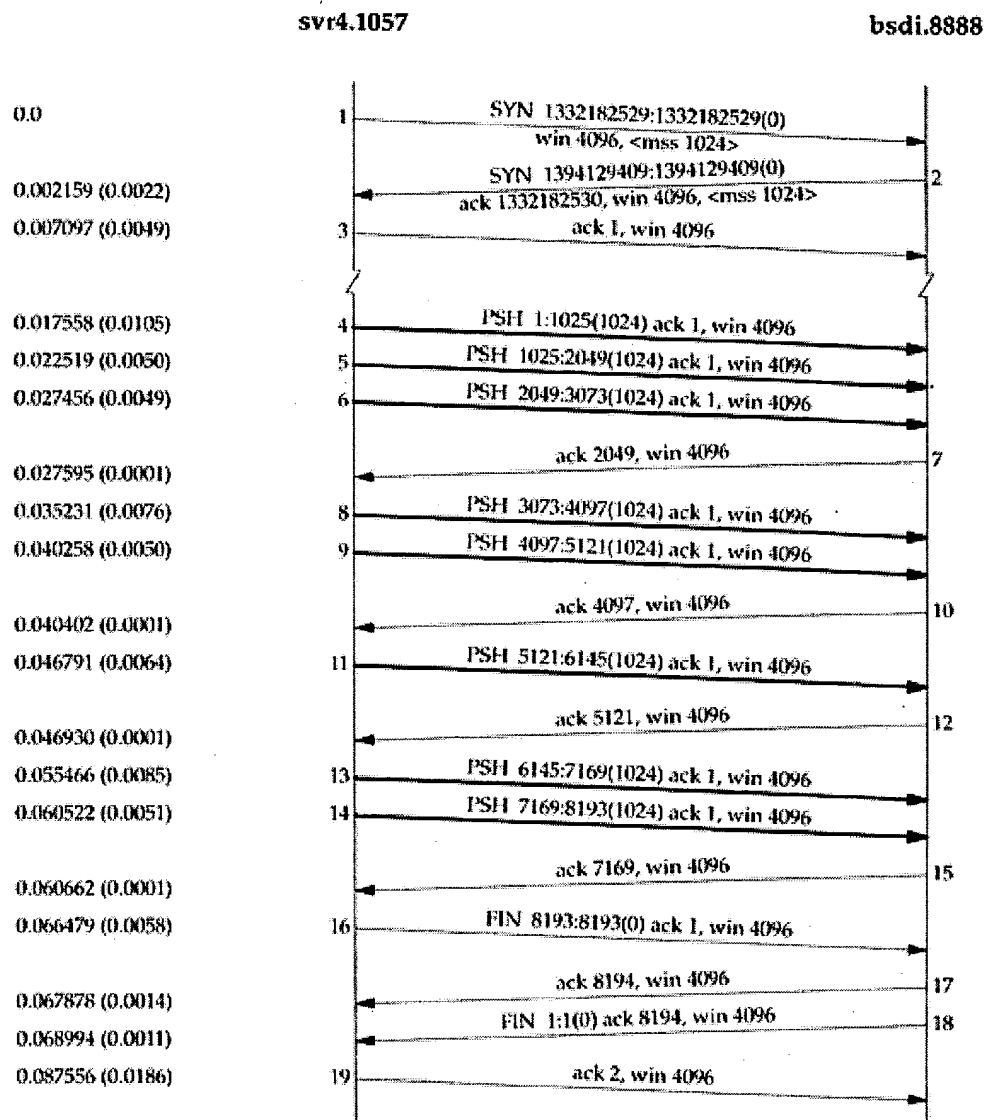
placed on IP's input queue. When segment 11 is processed by TCP the connection is marked for a delayed ACK. When segment 12 is processed, an ACK is generated (segment 14) for segments 11 and 12, and the delayed ACK flag for this connection is turned off. Segment 13 causes the connection to be marked again for a delayed ACK but before the timer goes off, segment 15 is processed, causing the ACK (segment 16) to be sent immediately.

It is important to notice that the ACK in segments 7, 14, and 16 acknowledge two received segments. With TCP's sliding-window protocol the receiver does not have to acknowledge every received packet. With TCP, the ACKs are cumulative—they acknowledge that the receiver has correctly received all bytes up through the acknowledged sequence number minus one. In this example three of the ACKs acknowledge 2048 bytes of data and two acknowledge 1024 bytes of data. (This ignores the ACKs in the connection establishment and termination.)

What we are watching with `tcpdump` are the dynamics of TCP in action. The ordering of the packets that we see on the wire depends on many factors, most of which we have no control over: the sending TCP implementation, the receiving TCP implementation, the reading of data by the receiving process (which depends on the process scheduling by the operating system), and the dynamics of the network (i.e., Ethernet collisions and backoffs). There is no single correct way for two TCPs to exchange a given amount of data.

To show how things can change, Figure 20.2 shows another time line for the same exchange of data between the same two hosts, captured a few minutes after the one in Figure 20.1.

Figure 20.2. Another transfer of 8192 bytes from `svr4` to `bsd`.

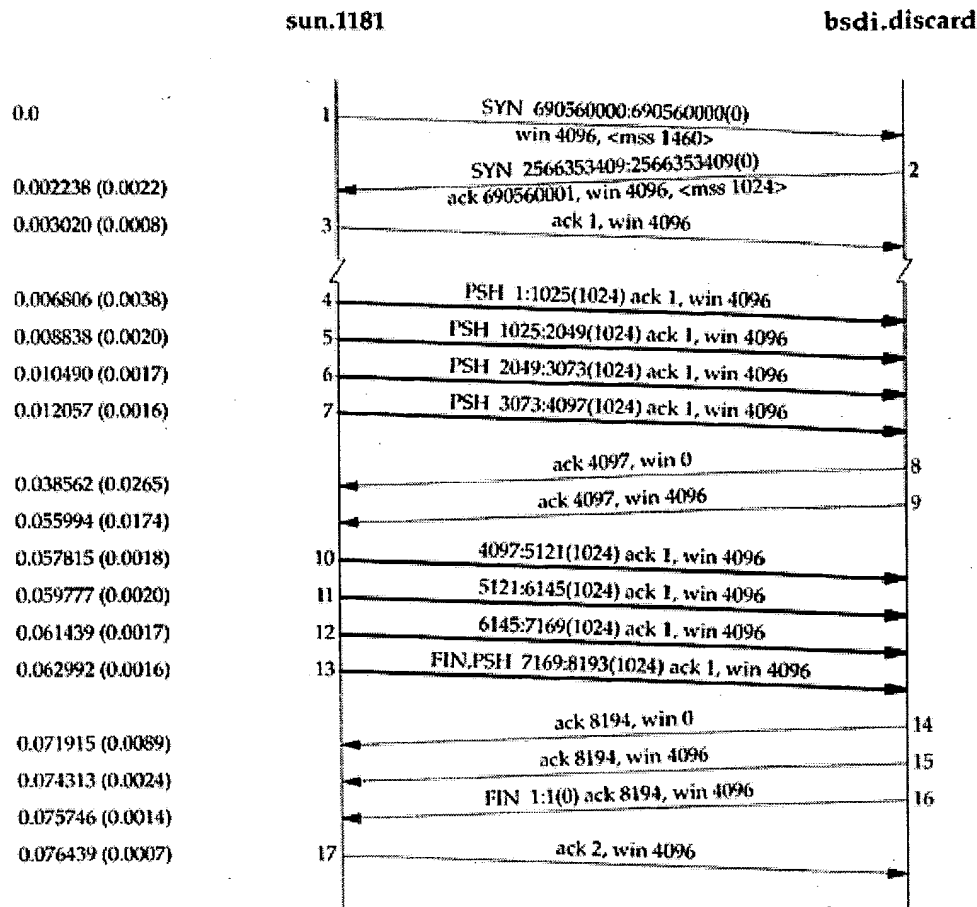


A few things have changed. This time the receiver does not send an ACK of 3073; instead it waits and sends the ACK of 4097. The receiver sends only four ACKs (segments 7, 10, 12, and 15): three of these are for 2048 bytes and one for 1024 bytes. The ACK of the final 1024 bytes of data appears in segment 17, along with the ACK of the FIN. (Compare segment 17 in this figure with segments 16 and 18 in Figure 20.1.)

Fast Sender, Slow Receiver

Figure 20.3 shows another time line, this time from a fast sender (a Sparc) to a slow receiver (an 80386 with a slow Ethernet card). The dynamics are different again.

Figure 20.3. Sending 8192 bytes from a fast sender to a slow receiver.



The sender transmits four back-to-back data segments (4–7) to fill the receiver's window. The sender then stops and waits for an ACK. The receiver sends the ACK (segment 8) but the advertised window is 0. This means the receiver has all the data, but it's all in the receiver's TCP buffers, because the application hasn't had a chance to read the data. Another ACK (called a *window update*) is sent 17.4 ms later, announcing that the receiver can now receive another 4096 bytes. Although this looks like an ACK, it is called a window update because it does not acknowledge any new data, it just advances the right edge of the window.

The sender transmits its final four segments (10–13), again filling the receiver's window. Notice that segment 13 contains two flag bits: PUSH and FIN. This is followed by another two ACKs from the receiver. Both of these acknowledge the final 4096 bytes of data (bytes 4097 through 8192) and the FIN (numbered 8193).

URL <http://proquest.safaribooksonline.com/0201633469/ch20lev1sec2>

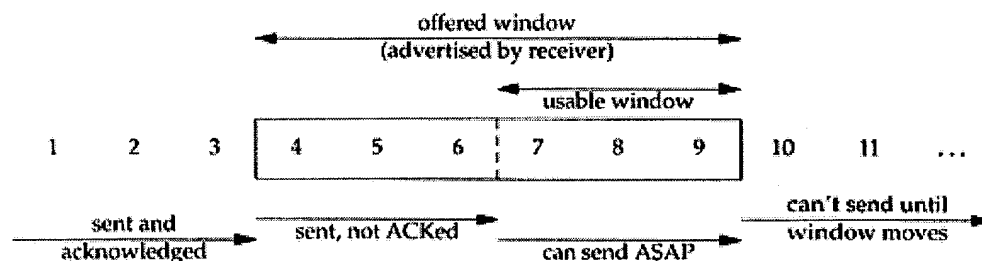
User name: US Patent and Trademark Library
Book: TCP/IP Illustrated, Volume 1: The Protocols
Section: Chapter 20. TCP Bulk Data Flow

No part of any chapter or book may be reproduced or transmitted in any form by any means without the prior written permission for reprints and excerpts from the publisher of the book or chapter. Redistribution or other use that violates the fair use privilege under U.S. copyright laws (see 17 USC107) or that otherwise violates these Terms of Service is strictly prohibited. Violators will be prosecuted to the full extent of U.S. Federal and Massachusetts laws.

20.3 Sliding Windows

The sliding window protocol that we observed in the previous section can be visualized as shown in Figure 20.4.

Figure 20.4. Visualization of TCP sliding window.



In this figure we have numbered the bytes 1 through 11. The window advertised by the receiver is called the *offered window* and covers bytes 4 through 9, meaning that the receiver has acknowledged all bytes up through and including number 3, and has advertised a window size of 6. Recall from Chapter 17 that the window size is relative to the acknowledged sequence number. The sender computes its *usable window*, which is how much data it can send immediately.

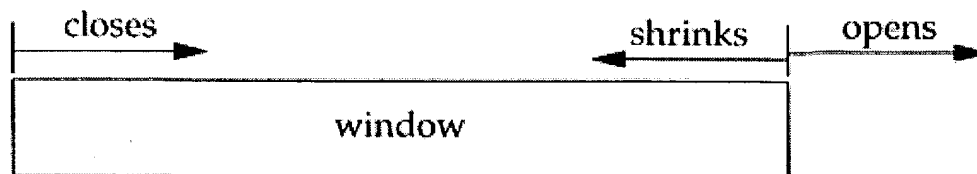
Over time this sliding window moves to the right, as the receiver acknowledges data. The relative motion of the two ends of the window increases or decreases the size of the window. Three terms are used to describe the movement of the right and left edges of the window.

1. The window *closes* as the left edge advances to the right. This happens when data is sent and acknowledged.
2. The window *opens* when the right edge moves to the right, allowing more data to be sent. This happens when the receiving process on the other end reads acknowledged data, freeing up space in its TCP receive buffer.
3. The window *shrinks* when the right edge moves to the left. The Host Requirements RFC strongly discourages this, but TCP must be able to cope with a peer that does this. Section 22.3 shows an example when one side would like to shrink the window by moving the right edge to the left, but cannot.

Figure 20.5 shows these three terms. The left edge of the window cannot move to the left, because this edge is controlled

by the acknowledgment number received from the other end. If an ACK were received that implied moving the left edge to the left, it is a duplicate ACK, and discarded.

Figure 20.5. Movement of window edges.

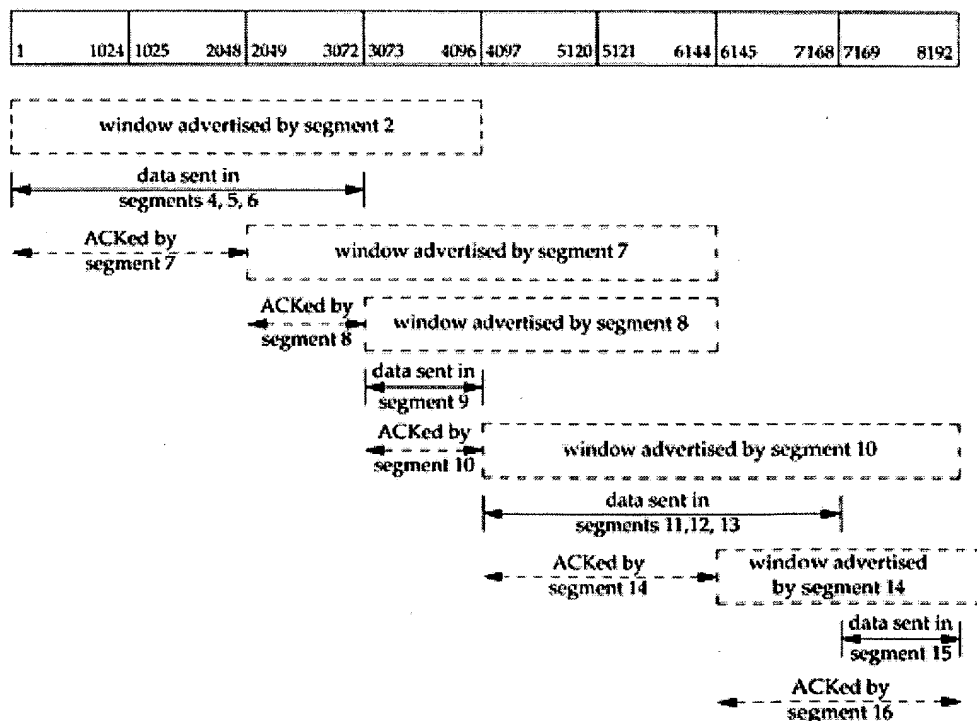


If the left edge reaches the right edge, it is called a *zero window*. This stops the sender from transmitting any data.

An Example

Figure 20.6 shows the dynamics of TCP's sliding window protocol for the data transfer in Figure 20.1.

Figure 20.6. Sliding window protocol for Figure 20.1.



There are numerous points that we can summarize using this figure as an example.

1. The sender does not have to transmit a full window's worth of data.
2. One segment from the receiver acknowledges data and slides the window to the right. This is because the window size is relative to the acknowledged sequence number.

3. The size of the window can decrease, as shown by the change from segment 7 to segment 8, but the right edge of the window must not move leftward.
4. The receiver does not have to wait for the window to fill before sending an ACK. We saw earlier that many implementations send an ACK for every two segments that are received.

We'll see more examples of the dynamics of the sliding window protocol in later examples.

URL <http://proquest.safaribooksonline.com/0201633469/ch20lev1sec3>



Welcome
United States Patent and Trademark Office



Welcome to IEEE Xplore®

- ☐ Home
- ☐ What Can I Access?
- ☐ Log-out

Tables of Contents

- ☐ Journals & Magazines
- ☐ Conference Proceedings
- ☐ Standards

Search

- ☐ By Author
- ☐ Basic
- ☐ Advanced

Member Services

- ☐ Join IEEE
- ☐ Establish IEEE Web Account
- ☐ Access the IEEE Member Digital Library

IEEE Enterprise

- ☐ Access the IEEE Enterprise File Cabinet



Your search matched **3** of **1075719** documents.

A maximum of **500** results are displayed, **15** to a page, sorted by **Relevance** in **Descending** order.

Refine This Search:

You may refine your search by editing the current search expression or entering a new one in the text box.

☐ Check to search within this result set

Results Key:

JNL = Journal or Magazine **CNF** = Conference **STD** = Standard

1 C-ICAMA, a centralized intelligent channel assigned multiple access for multi-layer ad-hoc wireless networks with UAVs

Gu, D.L.; Ly, H.; Xiaoyan Hong; Gerla, M.; Guangyu Pei; Yeng-Zhong Lee;
Wireless Communications and Networking Conference, 2000. WCNC. 2000
IEEE , Volume: 2 , 23-28 Sept. 2000
Pages:879 - 884 vol.2

[\[Abstract\]](#) [\[PDF Full-Text \(572 KB\)\]](#) IEEE CNF

2 Program generator for GPIB instruments

Akundi, I.; Smith, D.L.;
Southeastcon '97. 'Engineering new New Century', Proceedings. IEEE , 12-14 April 1997
Pages:52 - 57

[\[Abstract\]](#) [\[PDF Full-Text \(424 KB\)\]](#) IEEE CNF

3 Design and implementation of streaming system for MPEG-4 based interactive contents over IP networks

Hyun-Cheol Kim; Kyuheon Kim; Seunghong Min;
Digital and Computational Video, 2002. DCV 2002. Proceedings. Third International Workshop on , 14-15 Nov. 2002
Pages:100 - 107

[\[Abstract\]](#) [\[PDF Full-Text \(627 KB\)\]](#) IEEE CNF



US Patent & Trademark Office

[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide

end* and file* and status* and await* and request* and buffer*



THE ACM DIGITAL LIBRARY


[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

Terms used

end and file and status and await and request and buffer and response

Found 34,427 of 142,983

Sort results by

relevance

[Save results to a Binder](#)[Try an Advanced Search](#)[Try this search in The ACM Guide](#)

Display results

expanded form

[Search Tips](#)☐ Open results in a new window

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

Relevance scale ☐ ☐ ☐ ☐ ☐1 [Status report of the graphic standards planning committee](#)

Computer Graphics staff

August 1979 **ACM SIGGRAPH Computer Graphics**, Volume 13 Issue 3Full text available: pdf(15.01 MB) Additional Information: [full citation](#), [references](#), [citations](#)2 [Draft Proposed: American National Standard—Graphical Kernel System](#)

Technical Committee X3H3 - Computer Graphics

February 1984 **ACM SIGGRAPH Computer Graphics**, Volume 18 Issue SIFull text available: pdf(16.07 MB) Additional Information: [full citation](#)3 [PARLOG: parallel programming in logic](#)

Keith Clark, Steve Gregory

January 1986 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 8 Issue 1Full text available: pdf(3.79 MB) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

PARLOG is a logic programming language in the sense that nearly every definition and query can be read as a sentence of predicate logic. It differs from PROLOG in incorporating parallel modes of evaluation. For reasons of efficient implementation, it distinguishes and separates and-parallel and or-parallel evaluation. PARLOG relations are divided into two types: single-solution relations and all-solutions relations. A conjunction of single-solution relation calls can be evaluated ...

4 [Status report of the graphic standards planning committee of ACM/SIGGRAPH: State-of-the-art of graphic software packages](#)

Computer Graphics staff

September 1977 **ACM SIGGRAPH Computer Graphics**, Volume 11 Issue 3Full text available: pdf(9.03 MB) Additional Information: [full citation](#), [references](#)5 [The transport layer: tutorial and survey](#)

Sami Iren, Paul D. Amer, Phillip T. Conrad

December 1999 **ACM Computing Surveys (CSUR)**, Volume 31 Issue 4Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index](#)

Full text available:  [pdf\(261.78 KB\)](#)

[terms](#)


Transport layer protocols provide for end-to-end communication between two or more hosts. This paper presents a tutorial on transport layer concepts and terminology, and a survey of transport layer services and protocols. The transport layer protocol TCP is used as a reference point, and compared and contrasted with nineteen other protocols designed over the past two decades. The service and protocol features of twelve of the most important protocols are summarized in both text and tables. < ...

Keywords: TCP/IP networks, congestion control, flow control, transport protocol, transport service

6 The LACONIQ monitor: time sharing for online dialogues

Daniel L. Drew

December 1967 **Communications of the ACM**, Volume 10 Issue 12

Full text available:  [pdf\(1.02 MB\)](#)


Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

The LACONIQ (Laboratory Computer Online Inquiry) Monitor was developed primarily to support non-numerical applications such as retrieval from very large files by means of a "dialogue" between a system user and a retrieval application. The monitor was designed so that it could work with a small computer (an IBM System 360/30). Therefore techniques for resource allocation were important. For this reason the use of core storage, computational facilities, and input-output ...

7 The EAS-E application development system: principles and language summary

Harry M. Markowitz, Ashok Malhotra, Donald P. Pazel

August 1984 **Communications of the ACM**, Volume 27 Issue 8

Full text available:  [pdf\(1.37 MB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

EAS-E is based on the entity-attribute-set view of system description—a useful formalism for system modeling and planning even when programming is done in languages other than EAS-E.

Keywords: entity-attribute-set world view, entity-relation world view, hierarchical model, network model, relational model

8 Fast and flexible application-level networking on exokernel systems

Gregory R. Ganger, Dawson R. Engler, M. Frans Kaashoek, Héctor M. Briceño, Russell Hunt, Thomas Pinckney

February 2002 **ACM Transactions on Computer Systems (TOCS)**, Volume 20 Issue 1

Full text available:  [pdf\(500.67 KB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Application-level networking is a promising software organization for improving performance and functionality for important network services. The Xok/ExOS exokernel system includes application-level support for standard network services, while at the same time allowing application writers to specialize networking services. This paper describes how Xok/ExOS's kernel mechanisms and library operating system organization achieve this flexibility, and retrospectively shares our experiences and ...

Keywords: Extensible systems, OS structure, fast servers, network services


9 Distributed operating systems

Andrew S. Tanenbaum, Robbert Van Renesse

December 1985 **ACM Computing Surveys (CSUR)**, Volume 17 Issue 4

Full text available:

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


 pdf(5.49 MB)

[terms](#), [review](#)

Distributed operating systems have many aspects in common with centralized ones, but they also differ in certain ways. This paper is intended as an introduction to distributed operating systems, and especially to current university research about them. After a discussion of what constitutes a distributed operating system and how it is distinguished from a computer network, various key design issues are discussed. Then several examples of current research projects are examined in some detail ...

10 Human-computer interface development: concepts and systems for its management


H. Rex Hartson, Deborah Hix

March 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 1Full text available:  pdf(7.97 MB)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Human-computer interface management, from a computer science viewpoint, focuses on the process of developing quality human-computer interfaces, including their representation, design, implementation, execution, evaluation, and maintenance. This survey presents important concepts of interface management: dialogue independence, structural modeling, representation, interactive tools, rapid prototyping, development methodologies, and control structures. *Dialogue independence* is th ...

11 The V distributed system


David Cheriton

March 1988 **Communications of the ACM**, Volume 31 Issue 3Full text available:  pdf(2.55 MB)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

The V distributed System was developed at Stanford University as part of a research project to explore issues in distributed systems. Aspects of the design suggest important directions for the design of future operating systems and communication systems.

12 Computation and communication in R*: a distributed database manager


Bruce G. Lindsay, Laura M. Haas, C. Mohan, Paul F. Wilms, Robert A. Yost

February 1984 **ACM Transactions on Computer Systems (TOCS)**, Volume 2 Issue 1Full text available:  pdf(1.05 MB)Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

Keywords: distributed computation, distributed recovery protocols, site autonomy

13 Concepts and Notations for Concurrent Programming

Gregory R. Andrews, Fred B. Schneider

January 1983 **ACM Computing Surveys (CSUR)**, Volume 15 Issue 1Full text available:  pdf(4.02 MB)Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

14 The TRIPOS filing machine, a front end to a file server

M. F. Richardson, R. M. Needham

October 1983 **ACM SIGOPS Operating Systems Review , Proceedings of the ninth ACM symposium on Operating systems principles**, Volume 17 Issue 5Full text available:  pdf(771.11 KB)Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)


This paper discusses an experiment which sets out to improve the performance of a number of single user computers which rely on a general purpose file server for their filing systems. The background is described in detail in reference [1], but for completeness it is necessary to say something about it here. The Cambridge Distributed Computing System consists, at

the time of writing, of between 50 and 60 machines of various types, connected by a digital communications ring. On the ...

15 A proposal for certain process management and intercommunication primitives

Gary D. Knott

January 1975 **ACM SIGOPS Operating Systems Review**, Volume 9 Issue 1


Full text available:  [pdf\(1.79 MB\)](#) Additional Information: [full citation](#), [citations](#)



16 Performing remote operations efficiently on a local computer network

Alfred Z. Spector

April 1982 **Communications of the ACM**, Volume 25 Issue 4

Full text available:  [pdf\(1.58 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)



A communication model is described that can serve as a basis for a highly efficient communication subsystem for local networks. The model contains a taxonomy of communication instructions that can be implemented efficiently and can be a good basis for interprocessor communication. These communication instructions, called remote references, cause an operation to be performed by a remote process and, optionally, cause a value to be returned. This paper also presents implementation considerations ...

Keywords: communication models, efficient communication, transactions

17 WYLBUR: an interactive text editing and remote job entry system

Roger Fajman, John Borgelt

May 1973 **Communications of the ACM**, Volume 16 Issue 5

Full text available:  [pdf\(1.06 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#)




WYLBUR is a comprehensive system for manipulating all kinds of text, such as computer programs, letters, and manuscripts, using typewriter terminals connected to a computer. It has facilities for remote job entry and retrieval as well as facilities for text alignment and justification. A powerful method for addressing text by content is provided. This paper describes the external appearance of WYLBUR as well as its internal structure. A short description of the major features of ORVYL, a ge ...

Keywords: content addressing, data entry, document preparation, interactive terminal, interactive text editing, online text editing, program preparation, remote job entry, remote job retrieval, remote terminal, terminal, terminal system, text editing, time-sharing

18 A back-end computer for data base management

R. H. Canaday, R. D. Harrison, E. L. Ivie, J. L. Ryder, L. A. Wehr

October 1974 **Communications of the ACM**, Volume 17 Issue 10

Full text available:  [pdf\(864.08 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)



It is proposed that the data base management function be placed on a dedicated back-end computer which accepts commands (in a relatively high level language such as the CODASYL Data Base Task Group, April 1971 Report) from a host computer, accesses the data base on secondary storage, and returns results. The advantages of such a configuration are discussed. An experimental implementation, called the eXperimental Data Management System, XDMS, is described and certain conclusions about the ba ...

Keywords: Data Base Task Group Language, back-end computer, computer configurations, computer networks, data base management, data base portability, data base protection, information retrieval

19 Advanced use of Simula

Graham Birtwistle

January 1981 **Proceedings of the 13th conference on Winter simulation - Volume 1**


Full text available:  [pdf\(961.23 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

This paper is a tutorial on program development in Simula. It assumes a reading knowledge of Simula, and sketches the design of a local area network simulator (Cambridge Ring architecture) in five logical levels: machine interface, queueing, simulation primitives, data collection primitives and finally the network components. Besides program development technique, we also emphasize the value of class body actions, inner, the virtual mechanism and data protection.

20 Design of the real-time executive for the Univac(r) 418 system

John Michael Williams

January 1966 **Proceedings of the 1966 21st national conference**

Full text available:  [pdf\(1.06 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

The UNIVAC 418 system hardware The 418 is a small- to medium-scale real-time computer announced to the general public in September of 1964. It is available in two models, identical except for storage speed (two or four microseconds). Storage and registers

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)



US Patent & Trademark Office

[Subscribe \(Full Service\)](#) [Register \(Limited Service, Free\)](#) [Login](#)

 Search: ☒ The ACM Digital Library ☐ The Guide


THE ACM DIGITAL LIBRARY

[Feedback](#) [Report a problem](#) [Satisfaction survey](#)

Terms used

end and file and status and await and request and buffer and response

Found 34,427 of 142,983

Sort results by


[Save results to a Binder](#)
[Try an Advanced Search](#)
[Try this search in The ACM Guide](#)

Display results


[Search Tips](#)
☐ Open results in a new window

Results 21 - 40 of 200

 Result page: [previous](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

Best 200 shown

 Relevance scale ☐ ☐ ☐ ☐ ☐

21 [A large semaphore based operating system](#)

Søren Lauesen

 July 1975 **Communications of the ACM**, Volume 18 Issue 7

 Full text available: [pdf\(1.39 MB\)](#)

 Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

The paper describes the internal structure of a large operating system as a set of cooperating sequential processes. The processes synchronize by means of semaphores and extended semaphores (queue semaphores). The number of parallel processes is carefully justified, and the various semaphore constructions are explained. The system is proved to be free of "deadly embrace" (deadlock). The design principle is an alternative to Dijkstra's hierarchical structuring of operating system ...

Keywords: RC 4000, asynchronous structuring, buffering, cooperating processes, coroutines, correctness, deadlock, deadly embrace, debugging, hierarchical structuring, multiprogramming, operating system, operating system structure, parallel processes, program maintenance, program proving, project management, project planning, project scheduling, queue semaphores, reentrant code, reliability, semaphore applications, semaphores, software paging, synchronizing primitives, time schedule

22 [Experience Using Multiprocessor Systems—A Status Report](#)

Anita K. Jones, Peter Schwarz

 June 1980 **ACM Computing Surveys (CSUR)**, Volume 12 Issue 2

 Full text available: [pdf\(4.48 MB\)](#)

 Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

23 [Architecture, design, and implementation of a multimedia conference system](#)

Anna A. Hać, Dongchen A. Lu

 March 1997 **International Journal of Network Management**, Volume 7 Issue 2

 Full text available: [pdf\(517.69 KB\)](#)

 Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

In this article a new multimedia conference system is designed and implemented which allows a group of users to conduct a meeting in real time. Participants can jointly view and edit relevant multimedia information, including text, graphics, and still images distributed throughout the network. © 1997 John Wiley & Sons, Ltd.

24 [Some constraints and tradeoffs in the design of network communications](#)

E. A. Akkoyunlu, K. Ekanadham, R. V. Huber

November 1975

Proceedings of the fifth ACM symposium on Operating systems principles

Full text available:  [pdf\(787.30 KB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

A number of properties and features of interprocess communication systems are presented, with emphasis on those necessary or desirable in a network environment. The interactions between these features are examined, and the consequences of their inclusion in a system are explored. Of special interest are the time-out feature which forces all system table entries to "die of old age" after they have remained unused for some period of time, and the insertion property which states th ...

Keywords: Computer networks, Interprocess communication, Ports

25 Verifying Security

Maureen Harris Cheheyli, Morrie Gasser, George A. Huff, Jonathan K. Millen
September 1981 **ACM Computing Surveys (CSUR)**, Volume 13 Issue 3

Full text available:  [pdf\(4.68 MB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)


26 The Tyndall range control system: bringing network computing to C2 systems

Dan DeJohn
November 1994 **Proceedings of the conference on TRI-Ada '94**

Full text available:  [pdf\(1.24 MB\)](#) Additional Information: [full citation](#), [index terms](#)

27 A relational approach to monitoring complex systems


Richard Snodgrass
May 1988 **ACM Transactions on Computer Systems (TOCS)**, Volume 6 Issue 2

Full text available:  [pdf\(3.42 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#), [review](#)

Monitoring is an essential part of many program development tools, and plays a central role in debugging, optimization, status reporting, and reconfiguration. Traditional monitoring techniques are inadequate when monitoring complex systems such as multiprocessors or distributed systems. A new approach is described in which a historical database forms the conceptual basis for the information processed by the monitor. This approach permits advances in specifying the low-level data collection, ...


28 Transactions and synchronization in a distributed operating system

Matthew J. Weinstein, Thomas W. Page, Brian K. Livezey, Gerald J. Popek
December 1985 **ACM SIGOPS Operating Systems Review , Proceedings of the tenth ACM symposium on Operating systems principles**, Volume 19 Issue 5

Full text available:  [pdf\(974.32 KB\)](#) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

29 Hypervisor-based fault tolerance

Thomas C. Bressoud, Fred B. Schneider
February 1996 **ACM Transactions on Computer Systems (TOCS)**, Volume 14 Issue 1

Full text available:  [pdf\(1.89 MB\)](#) Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

Protocols to implement a fault-tolerant computing system are described. These protocols augment the hypervisor of a virtual-machine manager and coordinate a primary virtual machine with its backup. No modifications to the hardware, operating system, or application programs are required. A prototype system was constructed for HP's PA-RISC

instruction-set architecture. Even though the prototype was not carefully tuned, it ran programs about a factor of 2 slower than a bare machine would.

Keywords: fault-tolerant computing system, primary/backup approach, virtual-machine manager

30 Recovery guarantees for Internet applications

Roger Barga, David Lomet, German Shegalov, Gerhard Weikum

August 2004 **ACM Transactions on Internet Technology (TOIT)**, Volume 4 Issue 3

Full text available:  pdf(997.52 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

Internet-based e-services require application developers to deal explicitly with failures of the underlying software components, for example web servers, servlets, browser sessions, and so forth. This complicates application programming, and may expose failures to end users. This paper presents a framework for an application-independent infrastructure that provides recovery guarantees and masks almost all system failures, thus relieving the application programmer from having to deal with these f ...

Keywords: Exactly-once execution, application recovery, communication protocols, interaction contracts

31 MERT - a multi-environment real-time operating system

D. L. Bayer, H. Lycklama

November 1975 **Proceedings of the fifth ACM symposium on Operating systems principles**

Full text available:  pdf(968.55 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

MERT is a multi-environment real-time operating system for the Digital Equipment PDP-11/45 and 11/70 computers. It is a structured operating system built on top of a kernel which provides the basic services such as memory management, process scheduling, and trap handling needed to build various operating system environments. Real-time response to processes is achieved by means of preemptive priority scheduling. The file system structure is optimized for real-time response. Processes are bui ...

Keywords: Kernel, Operating system, Process, Supervisor

32 Managed file distribution on the universe network

Christopher S Cooper

June 1984 **ACM SIGCOMM Computer Communication Review , Proceedings of the ACM SIGCOMM symposium on Communications architectures and protocols: tutorials & symposium**, Volume 14 Issue 2

Full text available:  pdf(745.95 KB) Additional Information: [full citation](#), [abstract](#), [references](#), [index terms](#)

The file distribution system on the Universe Network consists of a distributed set of co-operating agents which provide clients with a reliable bulk file collection, transfer and delivery service. The agent systems incorporate specialised techniques for optimizing use of the satellite channel, as well as making available facilities for broadcast file distribution. The distributed system architecture and protocols are described, with emphasis on the separation of control and data transfer. A ...

33 Graphics Programming Using the Core System

R. Daniel Bergeron, Peter R. Bono, James D. Foley


December 1978 **ACM Computing Surveys (CSUR)**, Volume 10 Issue 4

Full text available:  pdf(2.92 MB) Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

34 DAD, the C.S.I.R.O operating system

B. J. Austin, T. S. Holden, R. H. Hudson

September 1967 **Communications of the ACM**, Volume 10 Issue 9

Full text available:  [pdf\(1.67 MB\)](#)

Additional Information: [full citation](#), [references](#), [index terms](#)

35 The SDC Time-Sharing System revisited

Jules I. Schwartz, Clark Weissman

January 1967 **Proceedings of the 1967 22nd national conference**

Full text available:  [pdf\(864.51 KB\)](#)

Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

In the light of accelerating interest, it seems worthwhile to review both the current status of TSS and some of the predictions made three years ago. Our review will include a brief overview of system changes, a discussion in some depth of resource allocation (which now appears to be the critical factor in general-purpose time-sharing systems), and some conclusions regarding how well our statements have withstood the test of time.

36 A programming environment for a timeshared system

Richard P. Gabriel, Martin E. Frost

April 1984 **Proceedings of the first ACM SIGSOFT/SIGPLAN software engineering symposium on Practical software development environments**, Volume 19, 9 Issue 5, 3

Full text available:  [pdf\(859.14 KB\)](#)


Additional Information: [full citation](#), [abstract](#), [references](#), [citations](#), [index terms](#)

In 1968 the Stanford Artificial Intelligence Laboratory began to construct a programming environment from a PDP-10, a pre-TOPS-10 DEC1 timesharing system, and some innovative terminal hardware. By now this has developed into a programming environment for a KL-10 that integrates our editor with various other system functions, especially the Lisp subsystem. We use the term 'SAIL' to refer to the Stanford A. I. Lab KL-10 computer running the WAITS timesharing system. [Ha ...

37 T: integrated building blocks for parallel computing

G. M. Papadopoulos, G. A. Boughton, R. Greiner, M. J. Beckerle

December 1993 **Proceedings of the 1993 ACM/IEEE conference on Supercomputing**


Full text available:  [pdf\(1.37 MB\)](#)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

38 Process Communication Based on Input Specifications

Jan van den Bos, R. Plasmeijer, Jan W. M. Stroet

July 1981 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 3 Issue 3


Full text available:  [pdf\(1.59 MB\)](#)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

39 Hypervisor-based fault tolerance

T. C. Bressoud, F. B. Schneider

December 1995 **ACM SIGOPS Operating Systems Review, Proceedings of the fifteenth ACM symposium on Operating systems principles**, Volume 29 Issue 5


Full text available:  [pdf\(1.26 MB\)](#)

Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

40 Data Communication Control Procedures

Byron W. Stutzman

December 1972 **ACM Computing Surveys (CSUR)**, Volume 4 Issue 4

Full text available:  [pdf\(1.36 MB\)](#)





Additional Information: [full citation](#), [references](#), [citations](#), [index terms](#)

Results 21 - 40 of 200

Result page: [previous](#) [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [next](#)

The ACM Portal is published by the Association for Computing Machinery. Copyright © 2004 ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact Us](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Media Player](#)  [Real Player](#)